Exam1 — ECE 438

April 20^{th}, 2020

Name:

David Kirby

- This is a open-book, open-note exam.
- You *must* work on this exam by yourself!
- Calculators are permitted, but no communication devices of any kind are allowed.
- Each question is marked with its number of points.
- Show your answers in the space provided for them. Write neatly and be well organized.
- Show your work if you want to get credit!

Good luck!

Problem	Maximum	Score
1	10	
2	25	
3	15	
4	20	
5	15	
6	15	
Total	100	

1. **(10 points)** Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3.0 GHz clock rate with a CPI of 1.5. P2 has a 2.8 GHz clock rate with a CPI of 1.2. P3 has a 1.6 GHz clock rate with a CPI of 0.8.

   (a) Which processor has the highest clock rate? Calculate the clock period of each processor.

   P1 has the highest clock rate @ 3.0 GHz

   clock period of each:

   $$P1\ \text{clock period} = \frac{1}{3.0\,GHz} = 333.33\,ps$$
   $$P2\ \text{clock period} = \frac{1}{2.8\,GHz} = 357.14\,ps$$
   $$P3\ \text{clock period} = \frac{1}{1.6\,GHz} = 625\,ps$$

   (b) Calculate the Million Instruction Per Second (MIPS) of each processor. Which one has the highest MIPS rating?

   $$MIPS = \frac{\text{clk rate}}{CPI} \cdot \frac{1}{10^6}$$

   $$MIPS_{P1} = \frac{3.0\,GHz}{1.5} \cdot \frac{1}{10^6} = 2000\ MIPS$$
   $$MIPS_{P2} = \frac{2.8\,GHz}{1.2} \cdot \frac{1}{10^6} = 2333.3\ MIPS$$
   $$MIPS_{P3} = \frac{1.6\,GHz}{0.8} \cdot \frac{1}{10^6} = 2000\ MIPS$$

   (c) The same benchmark is compiled for each of the processors such that P3 executes 18 million instructions to completion (i.e. has a dynamic instruction count of 20 million instructions), while P1 and P2 have a dynamic instruction count of 22.0 million instructions. Calculate the execution time of each processor? Which processor is fastest?

   $$t_{exec} = \frac{\text{dyn. instr.}}{\text{instr/sec}}$$

   $$t_{exec\,P1} = \frac{22\ \text{million instr}}{2000\ MIPS} = 11\,ms$$
   $$t_{exec\,P2} = \frac{22\ \text{million instr}}{2333.3\ MIPS} = 9.43\,ms$$ P2 is fastest
   $$t_{exec\,P3} = \frac{20\ \text{million instr}}{2000\ MIPS} = 10\,ms$$

   (d) Assuming the reference machine takes 83 milliseconds to execute the given benchmark, calculate the speedup of each processor compared to the reference machine.

   $$speedup_{P1} = \frac{83\,ms}{11\,ms} = 7.55\times$$
   $$speedup_{P2} = \frac{83\,ms}{9.43\,ms} = 8.80\times$$
   $$speedup_{P3} = \frac{83\,ms}{10\,ms} = 8.3\times$$

2. **(25 points)** Imagine you designed a small embedded processor and synthesized it for a 45nm process technology node with a target clock rate of 800MHz. During power analysis, you found that at the target clock rate with a supply voltage of 0.9V, this processor draws 1.4W of dynamic power and 0.1W of static power. Consider the following power and energy trade-offs:

   (a) Assuming a cryptographic operation takes 0.5 seconds to complete on your processor, what is the energy per cryptographic operation at the target clock rate?

$$P_{total} = P_{dynamic} + P_{static}$$
$$= 1.4W + 0.1W$$
$$= 1.5W$$

$$\text{Energy/op} = P_{total} \times t_{operation}$$
$$= 1.5W \times 0.5\ \text{sec/op}$$
$$= 0.75\ \text{J/op}$$

   (b) For certain applications, your processor performs cryptographic operations 4x faster than necessary. If you were to slow the clock down to 200MHz without adjusting the voltage, what would be the overall power draw? What would be the energy per cryptographic operation?

$$P_{dynamic} \propto CV^2f \rightarrow P_{dynamic_{new}} = 1.4W \cdot \frac{200MHz}{800MHz} = 0.35W$$

$$P_{total_{new}} = P_{dynamic_{new}} + P_{static} = 0.35W + 0.1W = 0.45W$$

$$t_{new} = 0.5s \cdot \frac{800MHz}{200MHz} = 2s$$

$$\text{Energy/op} = 0.45W \cdot 2s = 0.9\ \text{J/op}$$

   (c) Assuming you could safely drop the voltage to 0.6V when operating at a 200MHz clock, recalculate the power draw and energy per cryptographic operation. Assume the leakage *current* remains the same.

$$P_{dynamic} = 1.4W \cdot \frac{200MHz}{800MHz} \cdot \left(\frac{0.6V}{0.9V}\right)^2 = 0.156W$$

$$P_{static} = V \cdot I_{leakage} = 0.1W\left(\frac{0.6}{0.9}\right) = 0.067W$$

$$P_{total} = 0.156W + 0.067W = 0.222W$$

$$\text{Energy/op} = 0.222W \cdot 2s = 0.444\ \text{J/op}$$

(d) The above questions looked at how the power and energy can be changed by varying the clock rate and voltage of a processor. Modern processors change these parameters dynamically, using a technique called Dynamic Voltage Frequency Scaling (DVFS). What are some other techniques for reducing energy? Briefly describe these techniques.

Clock gating affects the dynamic power by disabling the clock in unused logic. This keeps static power the same (as in part B) so that static-powered elements such as memory maintain their state.

Power gating is more like part c in that it disables power from unused logic. This can power down entire cores and saves both dynamic and static power.

(e) The technology node that a particular processor is fabricated with can also affect the energy efficiency. Imagine that you resynthesized your processor at a 130nm process technology node with a 1.5V supply voltage. In order to maintain the same transistor count, you set your target clock rate to 200MHz. Assuming the leakage *current* remains the same and that the capacitance of the design approximately scales linearly with the feature size, calculate the dynamic and static power for your processor at the 130nm node. What is the energy per cryptographic operation at the 130nm node and how does this compare to that of the 45nm node?

$$P_{dynamic} = 1.4W \cdot \left(\frac{130nm}{45nm}\right)\left(\frac{1.5V}{0.9V}\right)^2\left(\frac{200MHz}{800MHz}\right) = 2.809W$$

$$P_{static} = 0.1W\left(\frac{1.5V}{0.9}\right) = 0.167W$$

$$P_{total} = 2.809W + 0.167W = 2.9753W$$

$$t_{new} = 0.5s \cdot \frac{800MHz}{200MHz} = 2s$$

$$\text{Energy/op} = 2.9753W \cdot 2s = 5.95\ J/op$$

$$\frac{5.95\ J/op}{0.75\ J/op} = 7.93x \text{ higher}$$

3. **(15 points)** A program that you would like to run on a MIPS based microprocessor has a dynamic instruction count of $1.0 * 10^6$ and can be divided up as follows:

Instruction	Rate of Occurence
beq/bne	10%
j	2%
jal	3%
jr	3%
addu	15%
sub	6%
lui	2%
sll	6%
slt/slti	8%
and/andi	5%
lw	22%
sw	12%
lb	4%
sb	2%

$\text{control}_{\text{instr}} = 18\%$

alu instr = 42%

load & stores = 40%

(a) Assuming the average CPI for the MIPS machine is 1.6 for loads and stores, 1.2 for control flow instructions, and 1.0 for ALU instructions, what is the overall average CPI for the instruction mix shown above?

$CPI_{total} = (1.6 \cdot 40\%) + (1.2 \cdot 18\%) + (1.0 \cdot 42\%) = 1.276$

(b) Assuming the machine has a clock rate of 1.5GHz, calculate the execution time of this program.

$$t_{exec} = \frac{cc/instr \cdot \#instr}{cc/s} = \frac{1.276\ cc/instr \cdot 1 \times 10^6\ instr}{1.5 \times 10^9\ cc/s} = 0.861ms$$

(c) If you design another machine that has a 1.6GHz clock rate with an average CPI of 1.3 for loads and stores, 1.2 for control flow instructions and 0.8 for ALU instructions, how much faster would the new machine be?

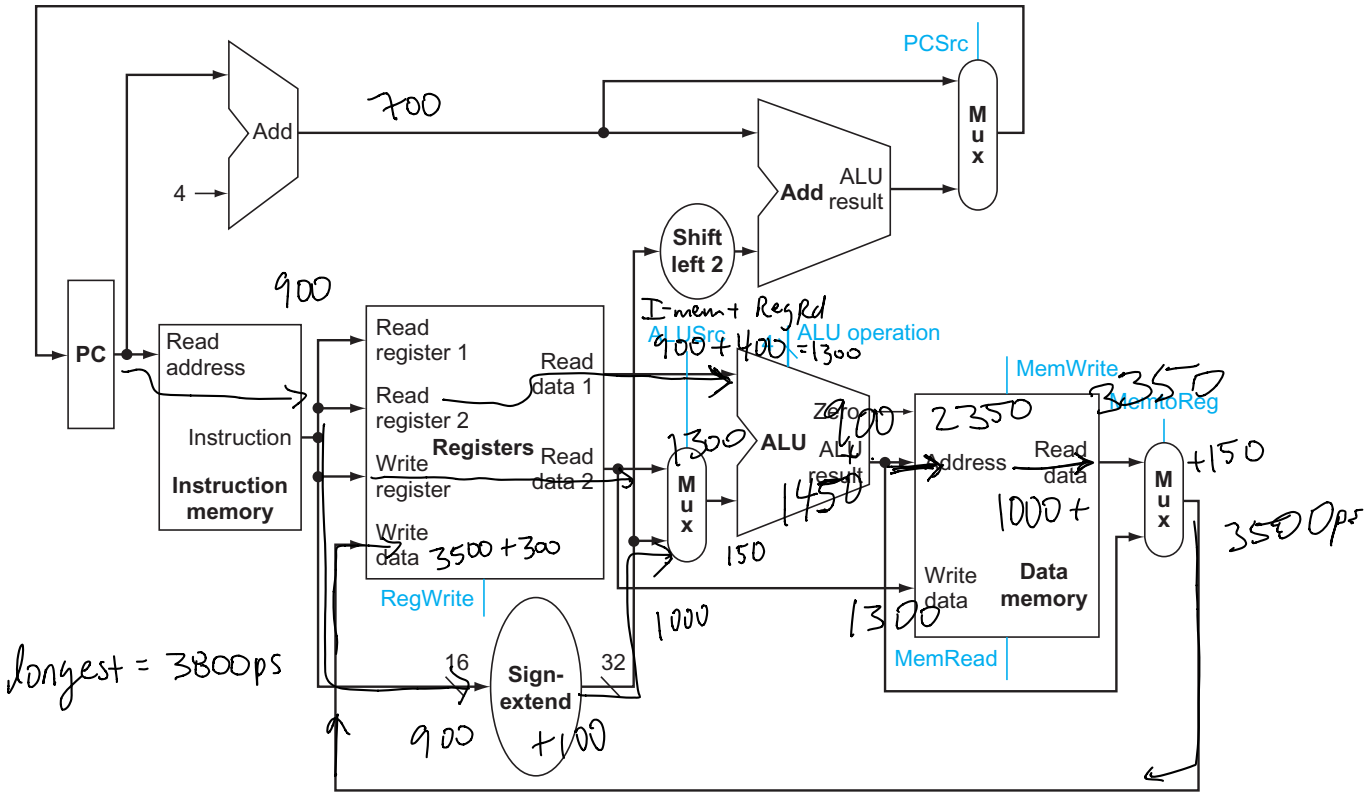
$$CPI_{total} = (1.3 \cdot 40\%)_{load} + (1.2 \cdot 18\%)_{ctrl} + (0.8 \cdot 42\%)_{alu} = 1.072$$

$$t_{exec} = \frac{cc/instr \cdot \#instr}{cc/s} = \frac{1.072\ cc/instr \cdot 1 \times 10^6\ instr}{1.6 \times 10^9\ cc/s} = 0.67ms$$

$$\text{speedup} = \frac{0.861ms}{0.67ms} = 1.2697x$$

4. **(20 points)** Consider the single-cycle datapath shown below with the following component latencies:

I-mem	Add	Mux	ALU	Reg Read	Reg Write	D-Mem	Sign-Extend	Shift-Left-2
900ps	700ps	150ps	900ps	400ps	300ps	1ns	100ps	75ps



(a) What is the maximum achievable clock rate for the datapath above? Ignore the effects of the control unit.
Hint: Mark the arrival time of valid data on each of the components in the figure above.

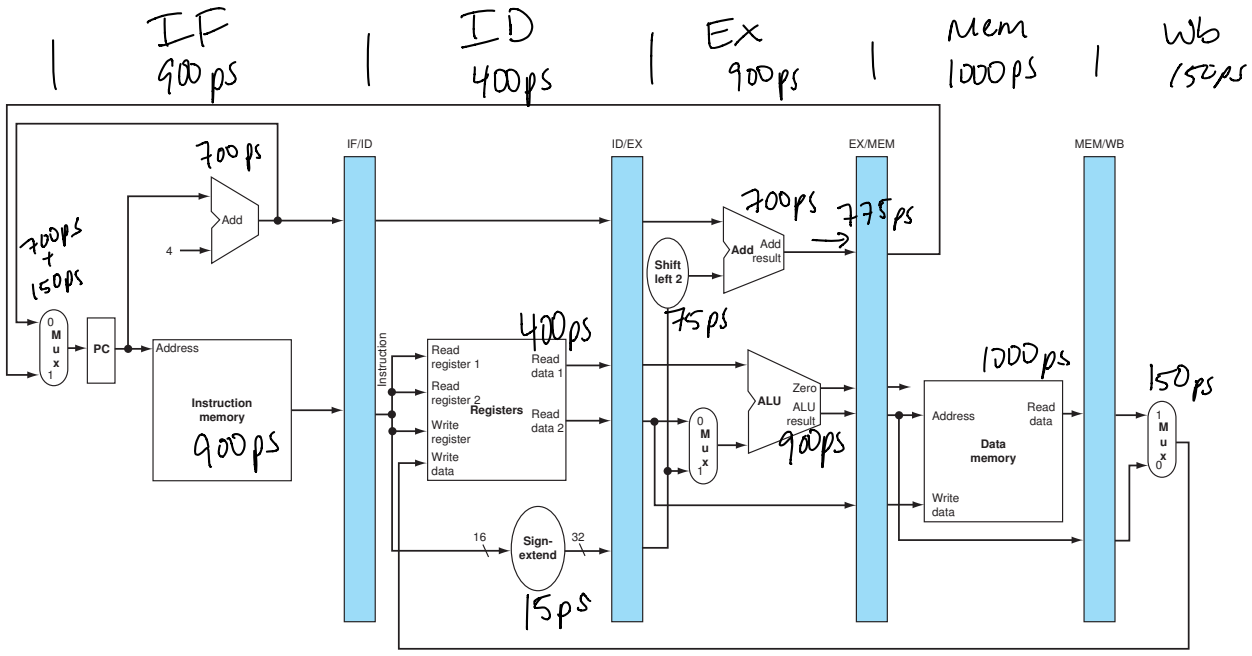
I-mem + Reg Rd + Mux + ALU + D-Mem + Mux + RegWrite
900ps + 400ps + 150ps + 900ps + 1000ps + 150ps + 300ps = 3800ps

$f = \frac{1}{3800ps} = 263.16\ MHz$

(b) Consider the pipelined datapath shown below. Ignoring the effects of the control unit and flip-flop delays, what is the maximum achievable clock rate?

The critical path is 1ns so the clock rate would be

$\frac{1}{1ns} = 1GHz$



(c) Assuming a long running program with no pipeline hazards, what is the instruction throughput of the above pipelined processor? What is the speedup compared to the single-cycle machine?

$$\frac{1\,GHz}{263.16\,MHz} = 3.8x$$

(d) Briefly describe some nonidealities that erode the performance of pipelining?

Pipeline hazards can erode performance. For example, structural hazards happen when instructions are competing for the same resources, data hazards exist when there are data dependencies between nearby instructions, and control hazards exist due to branches and jumps.

Also, pipelining leaves a lot of components idle while filling and draining.

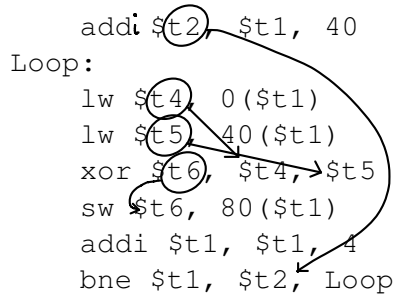
(e) If you were to add a 3-stage pipelined multiplier to your pipelined processor, such that each stage requires 900ps, calculate the speedup achieved by adding the multiplier assuming the dynamic instruction count is reduced by 72% and the CPI is increased by 11% with the addition of the multiplier.

Adding a 3-stage pipelined multiplier would bring our 5-stage pipeline to an 8-stage pipeline. Cycle time without the multiplier was 3800ps. Adding a 3-stage multiplier @ 900ps each stage adds 2700ps → 6500ps and a clock rate of 153.85MHz which is

$\frac{263.16\,MHz}{153.85\,MHz} = 1.71x$ slower. However, instruction count was reduced by 72% and CPI increased by 11% → $\frac{t_{new}}{t_{old}} = 1.71 \times 0.28 \times 1.11 = 0.531$

$$\text{speedup} = \frac{t_{old}}{t_{new}} = \frac{1}{0.531} = 1.88x$$

5. **(15 points)** Consider the code below:



```
     addi $t2, $t1, 40
Loop:
     lw $t4, 0($t1)
     lw $t5, 40($t1)
     xor $t6, $t4, $t5
     sw $t6, 80($t1)
     addi $t1, $t1, 4
     bne $t1, $t2, Loop
```

(a) Identify the RAW data dependency in the code above.

Please see above.

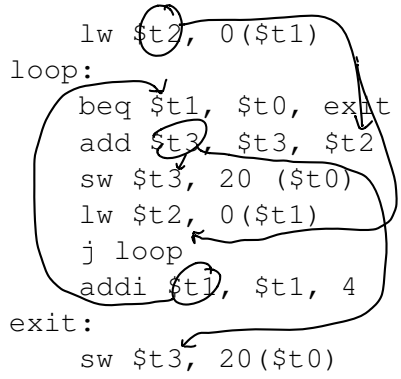
(b) Assuming a 5-stage pipelined processor without forwarding or hazard detection, rewrite the code above with `nops` to ensure correct execution. Also assume register write happens in the beginning of the clock cycle, and read happens at the end.

```
addi  $t2, $t1, 40
lw    $t4, 0($t1)
lw    $t5, 40($t1)
nop
nop
xor   $t6, $t4, $t5
nop
nop
sw    $t6, 80($t1)
addi  $t1, $t1, 4
bne   $t1, $t2, Loop
```

(c) Assuming a 5-stage pipelined processor with full data forwarding, hazard detection, and support for a single branch delay slot, reorder the code to eliminate as many `nops` as possible and make use of the branch delay slot.

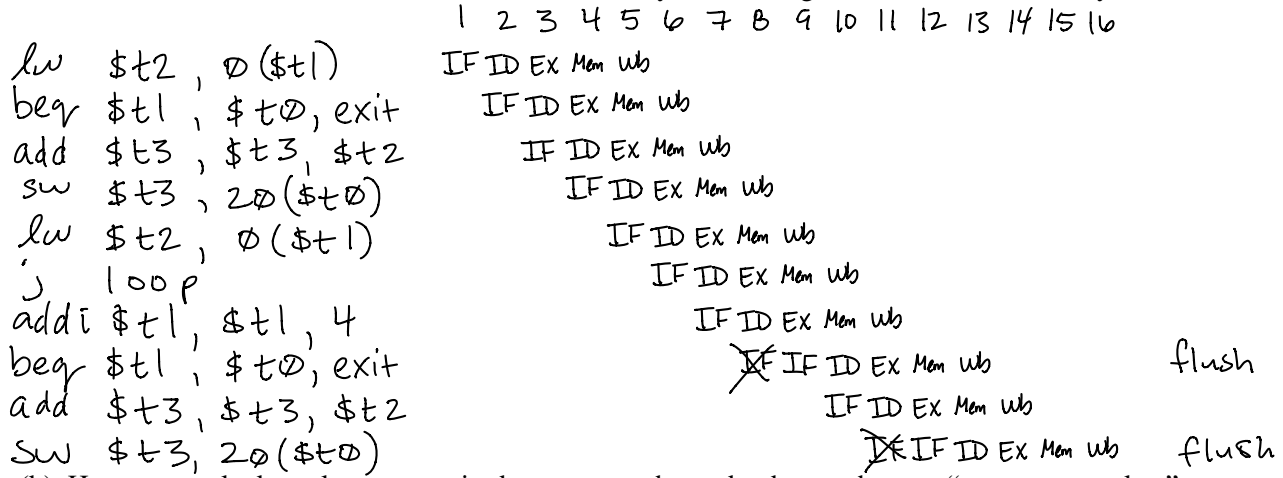
```
addi  $t2, $t1, 40
lw    $t4, 0($t1)
lw    $t5, 40($t1)
addi  $t1, $t1, 4
xor   $t6, $t4, $t5
bne   $t1, $t2, Loop
sw    $t6, 80($t1)
```

6. **(15 points)** For the following problem, assume a 5-stage pipelined processor with a branch delay slot and branch resolution in the Execute stage. Also assume the pipeline has full forwarding and hardware interlocking. Consider the code below:



```
    lw $t2, 0($t1)
loop:
    beq $t1, $t0, exit #not taken once, then taken
    add $t3, $t3, $t2
    sw $t3, 20 ($t0)
    lw $t2, 0($t1)
    j loop
    addi $t1, $t1, 4
exit:
    sw $t3, 20($t0)
```

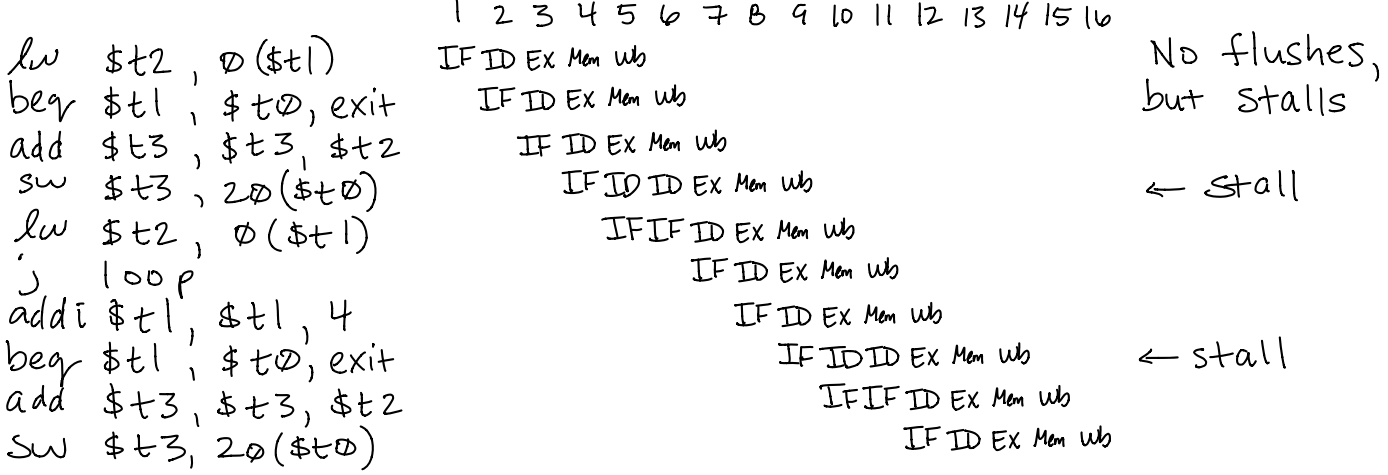
(a) Draw the pipeline execution diagram for the above code when an "assume not taken" branching scheme is used. Assume the code above has already been arranged to fill the branch delay slots.



(b) How many clock cycles are required to execute the code above when an "assume not taken" branching scheme is used?

16 clock cycles.

(c) If the branch resolution was moved to the Decode stage, how many clock cycles would be required? Draw the pipeline execution diagram.



So still 16 clock cycles.